

(NeXT Tip #54) Pretty printing C code

Christopher Lane (*lane[at]CAMIS.Stanford.EDU*)
Wed, 10 Aug 1994 14:29:59 -0700 (PDT)

There have always been various methods for 'pretty printing' or 'grinding' C-code for fancy printout. Typically formatting includes bolding of the language keywords, small italicized comments, variable width font, etc.

On our NeXTs There are currently three different ways that I'm aware of to produce nicely formatted printouts of code files:

- 1) vgrind
- 2) {ansic,c++,krc,objc}2latex
- 3) RTF source code

I'll comment a little on each of these as well as give some general notes on how to format your code to take advantage of them:

1) 'vgrind' is available on multiple Unix systems and is a csh script front end to 'troff' which formats several different programming languages using the parameters in /usr/lib/vgrindefs in combination with the troff macros defined in /usr/lib/tmac/tmac.vgrind -- typical usage is:

```
vgrind [-l<language>] <filename> [<filename> ...]
```

Where the -l option specifies what programming language is in the files with 'C' (or 'c') being the default. The output of vgrind is sent to your 'default' printer so make sure that you have a 'setenv PRINTER ...' command in your .login/.cshrc and that the printer listed is current/correct.

I created a vgrind definition for Objective-C that was adopted into the Berkeley Unix distribution of 'vgrindefs' a few years ago so the syntax and keywords of Objective-C are known to vgrind. Indicate that you are using Objective-C by using the -lObjC or -lobjc option.

One of the useful options to vgrind is the -x switch to create an index of functions across one or more files. However, it's a two pass process so see the 'man' page for vgrind for details. There are other options including one to make vgrind behave like a Unix filter.

I've used vgrind many times and it's a reasonable code formatter. The biggest problem I have with it is the way it handles long lines (which I use a lot) that don't fit the indenting and page width. Depending on the line itself, it will sometimes left justify it and other times put it in the correct place but write the extra characters on top of each other. In general, it's hard to convince vgrind to do anything other than what it wants to do. (E.g. try to get it to print in landscape rather than portrait.)

2) {ansic,c++,krc,objc}2latex are a group of new routines on /usr/local/bin recently submitted to USENET by Joerg Heitkoetter <joke[at]Germany.EU.net>. These routines convert various forms of C code (ANSI C, C++, old style K&R C &, of course, our friend Objective-C) into LaTeX documents.

The package defaults to formatting code snippets for insertion into larger documents. If you want to format a code file for its own sake, use the -c option -- you can use the routines as Unix filters:

```
> objc2latex -c < MyApplication.m > MyApplication.tex
```

Once you have the *.tex file, you can use the usual command line TeX routines to create a *.dvi and/or *.ps file for the printer.

Or, possibly more simply, open the *.tex file in the TeXview application, the default owner of *.tex files, where you can preview and print it. If you're

not familiar with TeXview, you'll want to set the 'Generate fonts as needed' option in the Preferences panel and set the 'Custom Resolution:' to 400 in the Print panel (don't use 600 as not all the TeX routines can handle it.)

For more details about embedding snippets of code and other options, see the man page for 'c++2latex' which covers all the routines.

3) RTF source code -- the trick here is not to pretty print your code on printout but instead write it pretty in RTF, rather than just plain ASCII, in the first place. This ability has been available for the NeXT on and off for the last several releases. The basic idea is that the compiler's preprocessor knows how to strip out the text from an RTF document, so you can format it as you see fit using Edit in rich text mode.

I only recommend this if you're going to make full use of it -- it isn't worth it for just bolding keywords and italicizing comments -- on NeXT-specific code. The added advantage of RTF source is that you can have hot links to man pages and other documentation, include illustrations in your comments, etc.

The largest application, that I'm aware of, written in RTF and publicly available is the NeXT GUI implementation of Archie by Scott Stark <stark[at]superc.che.udel.edu> which can be found in /LocalDeveloper/Examples. If you're interested in RTF source code, take a look at some of his *.m files and read the comments in his README file regarding RTF incompatibilities. (E.g. his *.h files are in ASCII to accomodate InterfaceBuilder.)

The only general guidelines I have regarding coding -- knowing that you're going to pretty print the code (or not) -- have to do with tabs and spaces:

a) You should always indent with tabs, not spaces. If you don't like how far the tab indents, change it in Edit's preferences (I've changed mine to be equivalent to 4 spaces). Most code you find is a mix of tabs and spaces that doesn't look good after being put through a pretty printer. (And is usually somewhat uneven after much cutting & pasting of code.)

b) Never use tabs for anything but indenting -- don't use them and multiple spaces for fancy formatting. For example, code formatted in ASCII like:

```
int          i;
NXRect      handFrame = { -25, -25, 10, 10 };
HandTileView *computerHand;
```

Usually look like:

```
int          i;
NXRect      handFrame = {    -25, -25, 10, 10 };
HandTileView *computerHand;
```

when pretty printed or even just viewed with a different size tabstop. Don't try to make comments or variables line up, it will always end up a mess. Usually, avoiding things that confuse a pretty printer will also make your code easier for someone else to edit, cut, copy & paste.

- Christopher

re: (NeXT Tip #54) Pretty printing C code

Christopher Lane (*lane[at]CAMIS.Stanford.EDU*)
Thu, 18 Aug 1994 09:25:18 -0700 (PDT)

> *On our NeXTs There are currently three different ways that I'm aware of to*
> *produce nicely formatted printouts of code files:*
>
> 1) *vgrind*
> 2) *{ansic,c++,krc,objc}2latex*
> 3) *RTF source code*

Turns out there's another: indent

Indent is a C program formatter. It reformats programs according to the nearly three dozen switches described in it's 'man' page. Although indent is intended for generating nicely formatted online code files, it has a 'troff' option which makes it format code for printing much like 'vgrind'.

Playing with indent, I've not been able to get it to make a program look better, or for that matter, even get it to not make a program look worse. This may have to do with my personal code formatting style being in conflict with its view of things -- you may have better luck.

The default switch settings will generate code that has some of the formatting problems I described earlier which cause other code formatters/printers to produce worse results. I recommend carefully tuning the options.

Also, 'indent' doesn't understand the // comment to end of line notation and not all options (e.g. '-st') can be moved into the ~/.indent.pro file. Like 'vgrind', the 'indent' program is available on other BSD-based Unix systems but has been augmented to understand Objective-C on the NeXT.

- Christopher